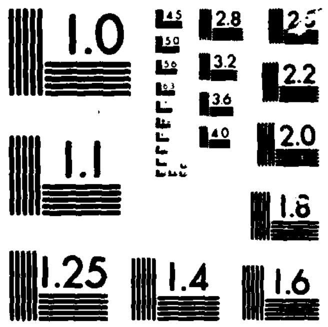
AD-A168 597   DESIGN OF A COMPUTER-AIDED INSTRUCTIONAL MODULE FOR THE   1/1
NAVY OCCUPATIONAL..(U) R-K RESEARCH AND SYSTEM DESIGN
MALIBU CALIF D M RAMSEY-KLEE ET AL. APR 86
UNCLASSIFIED   N00014-85-C-0813   F/G 6/14   NL



MICROCOPY CHART

(12)

R-K RESEARCH AND SYSTEM DESIGN

DESIGN OF A COMPUTER-AIDED INSTRUCTIONAL MODULE FOR THE NAVY OCCUPATIONAL HEALTH INFORMATION MANAGEMENT SYSTEM (NOHIMS)

PHASE I FINAL REPORT

Contract N00014-85-C-0813

April 1986

Diane M. Ramsey-Klee, Ph.D.
Principal Investigator
Donald D. Beck
Henry M. Halff, Ph.D.

The research reported here has been supported under Office of Naval Research contract N00014-85-C-0813 with funds provided by the Naval Medical Research and Development Command.

3947 RIDGEMONT DRIVE • MALIBU, CALIFORNIA 90265 • PHONE: (213) 456-6818

Technical Abstract

Phase I of Contract N00014-85-C-0813 involved five objectives or design steps: (1) selecting the best approach for designing a CAI module for NOHIMS, (2) specifying the data entry and retrieval functions in NOHIMS that constitute the subject matter for a CAI system, (3) identifying the knowledge required by a NOHIMS user, (4) defining the instructional approach and objectives of CAI for NOHIMS, and (5) determining the specific CAI functions to be implemented and the detailed CAI architecture.

In selecting an approach for NOHIMS CAI, two basic choices need to be made: (1) one needs to decide whether training should be offered as a part of NOHIMS itself or as a separate system using different hardware and/or system software, and (2) one needs to select a strategy for developing and delivering instruction. Currently there are three alternative strategies available---Intelligent Tutoring Systems; development of a computer program, written in a special CAI authoring language or a general-purpose programming language, that directly controls the instructional interaction with the student; and use of an existing authoring system to provide a frame-based CAI system. There are possibilities for implementing any of these three approaches both within and outside the NOHIMS environment. Consequently, there are six major alternatives that need to be considered.

When these six alternatives were evaluated and compared in terms of instructional features (the prime determinants of the system's effectiveness) and development features (the prime determinants of system cost), the following findings emerged. With respect to instructional effectiveness, frame-based approaches can be eliminated because they fail to provide practice which is considered to be an essential feature of any CAI system for users of information management systems. If, in addition, the CAI system is to be fully integrated with NOHIMS' hardware and software, only CAI programs written in MUMPS are viable alternatives. Further, if a major overhaul of the CAI software becomes necessary because of changes to NOHIMS, only Intelligent Tutoring Systems that are generic to MUMPS application systems remain in the competition. Such a tool would be adaptable not only for NOHIMS but for any other MUMPS-based system where computer-aided instruction could augment or replace traditional classroom instruction and user manuals.

None of the existing CAI software~~, either those packages written in MUMPS or in other languages,~~ provides the adaptable, generic capability needed to develop modifiable, interactive instructional software. If the NOHIMS CAI system architecture is to be applicable to a wide range of system applications, the alternative regarded as best is ~~the one designed in Phase I of this contract, namely,~~ the MUMPS-based ABC system---an Intelligent Tutoring System. ABC consists of three major executable components named ABLE, BILL, and CARLA. ABLE is used to specify the structures (data, file, and procedure) for the application, descriptive textual information needed for instruction, and curriculum structures. BILL is used to specify how the system behaves and to define the content of particular exercises. CARLA delivers instruction in the application's environment according to the specifications provided through ABLE and BILL. CARLA is the only component that needs to run in the application system's environment once preparation of the CAI curriculum, teaching examples, and practice exercises are completed. Development of the ABC system has the potential to impact importantly on how training for MUMPS-based systems will be delivered in the future.

For

A-1

Identification and Significance of Phase I Work

On September 23, 1985, R-K Research and System Design was awarded a Phase I contract by the Navy Medical Research and Development Command, NMC, NCR, to design a Computer-Aided Instruction (CAI) module for the Navy Occupational Health Information Management System (NOHIMS), to be funded through the FY 1985 Defense Small Business Innovation Research (SBIR) program. Phase I involved five objectives or design steps: (1) selecting the best approach for designing a CAI module for NOHIMS, (2) specifying the data entry and retrieval functions in NOHIMS that constitute the subject matter for a CAI system, (3) identifying the knowledge required by a NOHIMS user, (4) defining the instructional approach and objectives of CAI for NOHIMS, and (5) determining the specific CAI functions to be implemented and the detailed CAI architecture.

In selecting an approach for NOHIMS CAI, two basic choices need to be made: (1) one needs to decide whether training should be offered as a part of NOHIMS itself or as a separate system using different hardware and/or system software, and (2) one needs to select a strategy for developing and delivering instruction. Currently there are three alternative strategies available---Intelligent Tutoring Systems; development of a computer program, written in a special CAI authoring language or a general-purpose programming language, that directly controls the instructional interaction with the student; and use of an existing authoring system to provide a frame-based CAI system. There are possibilities for implementing any of these three approaches both within and outside the NOHIMS environment. Consequently, there are six major alternatives that need to be considered.

When these six alternatives were evaluated and compared in terms of instructional features (the prime determinants of the system's effectiveness) and development features (the prime determinants of system cost), the following findings emerged. With respect to instructional effectiveness, frame-based approaches can be eliminated because they fail to provide practice which is considered to be an essential feature of any CAI system for users of information management systems. If, in addition, the CAI system is to be fully integrated with NOHIMS' hardware and software, only CAI programs written in MUMPS are viable alternatives. Further, if a major overhaul of the CAI software becomes necessary because of changes to NOHIMS, only Intelligent Tutoring Systems that are generic to MUMPS application systems remain in the competition. Such a tool would be adaptable not only for NOHIMS but for any other MUMPS-based system where computer-aided instruction could augment or replace traditional classroom instruction and user manuals.

There are four distinct classes of NOHIMS users, and each class uses a different constellation of the data entry and retrieval functions of NOHIMS. Data entry clerks are responsible for accurately entering medical data and survey data into NOHIMS. Occupational health medical personnel use NOHIMS to obtain reports of medical data. Most physicians will use NOHIMS to retrieve data on individual patients under their care. Some physicians may also use NOHIMS to conduct miniature epidemiological investigations at their facility. Industrial hygiene/safety specialists use NOHIMS to track and monitor occupational hazards at Navy industrial sites and to generate ad hoc reports using an interactive query function. The NOHIMS system manager maintains system security; monitors data entry procedures for accuracy; produces management reports on a periodic basis; identifies any program, disk, or database errors so that corrective action can be taken; secures the system against data loss and restarts operations after a system crash; initiates back-up procedures on a daily basis; and is responsible for periodic purging and/or archiving of data files.

NOHIMS users need to know about the data structures present in NOHIMS, the files used to maintain data, and the procedures for data entry and retrieval. Each of these domains can be represented as a hierarchy. The knowledge requirements of individual users will vary widely among and within classes of NOHIMS users.

The instructional approach and objectives of CAI for NOHIMS are constrained by a number of aspects of the training environment. NOHIMS user training needs to be geared to situations with reasonably high personnel turbulence, which means that training should be available as a turnkey system. Some users, such as data entry clerks, will need to meet certain levels of competence before being given responsibility for data entry in the "live" system. Provision needs to be made for casual users of NOHIMS where high skill levels and thorough knowledge of the system are not particularly critical or appropriate. Since user reference manuals are often intimidating in size, lacking direct answers to particular questions, or missing from the physical vicinity of the terminals where they are needed, NOHIMS training (and operation) should be available in a paperless environment independent of external reference or training materials. In particular, users need a way of obtaining information on new features and changes as they are introduced into the system.

NOHIMS training must have the following features in order to be successful. Training should be offered in the operational NOHIMS environment, that is, a NOHIMS user should be able to use training facilities without terminating his or her interaction with NOHIMS. Practice should be the major component of an adequate training system offering a number of exercises on each NOHIMS procedure. Instruction should be given under the control of a curriculum. Students should have comprehensive reference material available on-line during both their interaction with NOHIMS and the CAI system, accessible via free-form textual inquiries.

If the design approach for implementing the NOHIMS CAI system architecture is to be applicable to a wide range of system applications, the alternative that we regard as best is the one we have designed in Phase I of this contract, namely, the MUMPS-based ABC system---an Intelligent Tutoring System. ABC consists of three major executable components named ABLE (Application Based Lexicon Engine), BILL (Base-line Instructional Lesson Literator), and CARLA (Conversationally Activated Reference and Lesson Aid). ABLE is used to specify the structures (data, file, and procedure) for the application, textual information needed for instruction, and curriculum structures. The second component, BILL, is used to specify how the system behaves and to define the content of particular exercises. The third component, CARLA, delivers instruction in the application's environment according to the specifications provided through ABLE and BILL. CARLA is the only component that needs to run in the application system's environment once preparation of the CAI curriculum, teaching examples, and practice exercises are completed.

Detailed Design Considerations Identified in Phase I

In this section we present our findings resulting from the five design steps followed in Phase I.

Step 1: Select an Instructional Approach

In documenting this step, we present the results of the process used to select the most promising CAI approach for NOHIMS training. The process involved delineating the different alternative approaches, defining the criteria for evaluating each of them, and finally, conducting a comparison.

Alternative Approaches

In selecting an approach for NOHIMS CAI, two basic choices need to be made. First, one needs to decide whether training should be offered as a part of NOHIMS itself or as a separate system using different hardware and/or system software. Second, one needs to select a strategy for developing and delivering instruction.

Currently there are three alternative approaches for developing and delivering instruction. One is the genre of CAI systems known as Intelligent Tutoring Systems (ITS) described in Sleeman and Brown (1982) and in Anderson, Boyle, and Reiser (1985). These systems use a high-level specification of the material to be taught and the activities needed to learn that material to generate individualized tutorial instruction. The second general approach to instruction is to develop a computer program, written in a special CAI authoring language or a general-purpose programming language, that directly controls the instructional interaction with the student. A third approach is to use an existing authoring system to provide a frame-based CAI system. Frame-based CAI uses the computer to deliver programmed instruction. That is, it presents material to students, tests their knowledge, and dynamically sequences these presentations and tests under the direction of a curriculum. Many of the issues relevant to these latter two alternatives can be found in Avner, Smith, and Tenczar (1984) and in Merrill (1985).

In laying out the alternatives corresponding to these basic choices, we found that no ITSs are available for the kinds of instruction needed by NOHIMS users. However, our system design that is presented in detail in the Description of the ABC System section of this report constitutes an ITS and can run under NOHIMS. As defined below, we are prepared to offer two different versions of ABC. Or alternatively, a CAI system could be coded directly in MUMPS and integrated with NOHIMS. Two frame-based instructional systems written in MUMPS called STUDY and DIALOG also exist that might be able to run under NOHIMS. And, of course, there are several possibilities for implementing any of these three approaches outside of the NOHIMS environment. Consequently, there are six major alternatives that need to be considered.

ABC---An Intelligent Tutoring System

The alternative that we regard as best is the ABC system. Since it is described in greater detail elsewhere in this report, here we provide only a summary of its major features for the purposes of evaluation.

ABC comes in two parts, an instructional system and a development system by which the instructional system is generated. ABC's instructional system (called CARLA) includes a free-form inquiry mode whereby the student can access descriptions of the structure and content of the application being taught (in this case, NOHIMS). CARLA also offers exercises and examples within a structured curriculum describing the procedures to be learned.

Development is supported by two ABC components. One, called ABLE, allows subject matter experts to develop structured descriptions of the data and procedures making up the application. The other, called BILL, allows subject matter experts to describe the behavior of the application and to use that description to build any number of exercises and examples to be offered to students. (Some examples will also be available in ABLE as part of the descriptive material.) All three modules will be fully integrated with each other and with the application.

Depending on the level of funding available, R-K Research and System Design is prepared to offer either of two versions of the system---ABC or AbC. We use "ABC" to denote the full system as described above. All three components would be provided. ABLE and BILL would be available for authoring instruction on new applications and for maintaining instructional materials on existing applications. We use the term "AbC" to refer to an incomplete version of ABC that could be provided if funding is unavailable for the complete system. AbC includes complete versions of ABLE and CARLA. BILL, however, would be replaced with MUMPS programs to control examples and very limited exercises. Using ABLE, the descriptive material could be easily changed by any subject matter expert (including examples incorporated into the descriptive material), but any exercises in BILL would have to be reprogrammed to accommodate changes in NOHIMS.

Conventional CAI Running with NOHIMS

There are two other ways to provide computer-assisted instruction that would run under NOHIMS.

Nongeneric CAI in MUMPS and/or in MUMPS PILOT.

It would, of course, be possible to provide all of the functionality of CARLA by directly coding the instructional materials in an appropriate programming language. Such a program would provide adequate CAI for the particular version of NOHIMS at hand, but any changes in the application or in the instructional materials would require reprogramming the CAI system.

MUMPS is one obvious choice as a language for this alternative since NOHIMS is a MUMPS application. Another possibility is a programming language known as PILOT, which is available under MUMPS as a public domain program called MUMPS PILOT. PILOT is a fully functional computer language that has special primitives for CAI applications. One example of such a primitive is a match operation that finds acceptable matches between answer keys and free-form input from the student.

Unfortunately, PILOT offers no benefits in this particular case. Few if any programmers are familiar with both NOHIMS and PILOT. Research (Avner, Smith, & Tenczar, 1984) indicates that authoring languages offer no particular advantage to skilled programmers over general-purpose languages. In addition, conventions for student interactions with PILOT conflict somewhat with corresponding NOHIMS conventions, raising the possibility of negative transfer. But most troublesome is the fact that MUMPS PILOT is not a fully functional version of the language (but rather a simple demonstration program) and has nowhere near the power needed to develop an adequate CAI for NOHIMS.

Frame-Based CAI. We know of two proprietary frame-based CAI systems that run under MUMPS---STUDY and DIALOG. While one or perhaps both of them might run on the same MUMPS systems that serve NOHIMS, it is not clear that they could be fully integrated to run as tasks under NOHIMS. Since frame-based CAI systems operate like turning the pages of a book, neither STUDY nor DIALOG could provide the ad hoc reference capability that CARLA makes available to users.

STUDY is a frame-based authoring system written in MUMPS developed and used by a firm called Interactive Learning Systems, Bolton, Massachusetts. STUDY allows authors to define a curriculum and then create instructional frames that are dynamically sequenced by the curriculum. Systems such as this are appropriate for delivering descriptive material and testing students' knowledge of that material, but they are not suited to hands-on instruction. In addition, STUDY is proprietary and not available to the public for instructional development. If the Navy wanted to pursue this alternative, they would have to contract with Interactive Learning Systems to develop and maintain the CAI system for them.

DIALOG is another frame-based authoring tool running under MUMPS. It was developed at Massachusetts General Hospital (MGH) in Boston as an automated medical history questionnaire originally. Over the last several years this authoring system has been substantially improved and is used in health education at MGH to produce basic tutorials and clinical simulations. The authoring side of DIALOG employs a multiple branching logic to present frames of instructional material to students. There are two execution points per frame where the author (if he or she is a MUMPS programmer) may insert executable MUMPS code. Certain frames can be subroutines, and DIALOG itself can be called as a subroutine from a MUMPS application system. Some free-form text can be input by a student, but this capability is limited. It offers none of the deep interactive features needed for providing practice exercises with NOHIMS.

DIALOG exists in two versions of MUMPS---in MGH MUMPS (the dialect in use at MGH) and more recently in ANSI-Standard MUMPS. To date documentation has been prepared for the MGH MUMPS version only. DIALOG has never been licensed for use outside the MGH environment although it may become available in the near future.

Conventional CAI Running outside NOHIMS

Some advantage might be gained from delivering instruction using a microcomputer or some other hardware system external to NOHIMS. The intrinsic graphics capability of microcomputers facilitates a more powerful presentation mode. Certainly the student interface could be improved by these features, and students could view instruction on one display while practicing with NOHIMS on another. In addition, a wider range of computer languages and authoring systems is available on non-MUMPS systems.

On the other hand, these alternatives would be costly. Furthermore, they would require students to learn about one computer (the CAI system) before they could learn about the application system (NOHIMS) since there would be no similarity in surface behavior between the two systems. Additionally, there would be no operational continuity or computational link between the two systems so that instruction could not be informed by performance.

Nongeneric CAI. It is theoretically possible to program a fully functional CAI system outside MUMPS, just as could be done with MUMPS or fully functional MUMPS PILOT. Several special authoring languages (e.g., TUTOR, PILOT, EnBASIC) are available for this purpose. In addition, a general-purpose programming language such as PASCAL could be used. Although such languages provide complete flexibility in designing instruction, they usually entail the enormous development and maintenance costs associated with large software systems.

Frame-Based CAI Not Written in MUMPS. Frame-based CAI systems such as TICCIT are also available on many computers and could be used to deliver purely descriptive information about NOHIMS and its operating procedures. No particular programming expertise is needed to bring up instruction using these systems, but the range of instructional methods is severely limited.

Bases for Evaluation

We evaluated the approaches described above in terms of two general criteria. We first looked at instructional features, that is, what is provided to students in the way of instruction. These features are the prime determinants of the system's effectiveness. We then looked at development features, that is, what is involved in developing and maintaining the system. These features are the prime determinants of system cost.

Instructional Features

There are two major factors to consider in looking at instructional effectiveness. One has to do with the training environment and, in particular, whether training is to be offered in the NOHIMS environment or not. The other major factor concerns the kind of instruction available.

Levels of Integration with NOHIMS. The systems that we have discussed can be integrated with NOHIMS at any of three increasing levels of compatibility. At the lowest level (noncompatibility), the CAI system and NOHIMS operate as completely separate systems with regard to both hardware and software. At the intermediate level, the CAI system and NOHIMS operate on the same computer but as separate software systems. No new hardware buys are necessary, but the user must switch out of the NOHIMS context to get instruction. At the highest level of compatibility, the CAI system runs under the NOHIMS software on the same computer. Instruction is available to users while they are interacting with NOHIMS.

Instructional Functionality. The major distinction that needs to be made with respect to instructional functionality concerns the availability of practice. We therefore talk about two different types of instructional methods---descriptive methods that tell the student about the application and methods for guided practice in its use.

All of the systems discussed above can make descriptive material available to the student. In a typical CAI system, the material can be delivered in an interactive fashion under the control of a curriculum. In addition, tests can be given to determine progress through the curriculum.

We consider practice to be an essential feature of any CAI system for users of information management systems. In our case, students should be able to work with NOHIMS under practice conditions. Ideally, they should be able to obtain context-sensitive help when they run into trouble and they should be provided with worked out examples of selected exercises. Exercises, like descriptive material, should be selected and sequenced by a curriculum.

Development Features

The facilities available to develop and maintain a CAI system will be major determinants of its costs. The questions that need to be asked about such facilities concern the structure of the development system and the entry-level skills for using it.

Modularity and Structure. There are two general ways that the structure of CAI development systems can facilitate the implementation of specific CAI modules. First, the development system can allow for high-level specification of particular instructional regimes. Second, the development system may allow for a high-level specification of the subject matter (in this case, NOHIMS) for use in specifying the regimes.

Support for Instruction Specification. Ideally, a CAI development system should know enough about instruction that developers need only specify the particulars of the instructional regime needed for their applications. Development systems can provide this kind of support for four aspects of instruction. At the least, a system should implement the details of a consistent student interface. That is, it should make available pre-specified testing formats, handle spelling correction and other input pre-processing, and implement standard options such as on-line help. Development systems can also allow for separate specification of the verbal descriptions needed for instruction and tutoring under the CAI system. Beyond this, many systems allow for specification of a curriculum or prerequisite hierarchy that is used to select and sequence instructional events. Most difficult are provisions that allow for the abstract specification of exercises or examples that can be provided to the student under tutorial conditions.

Support for Subject Matter Specification. The ideal CAI system would know enough about instruction so that it could manufacture instruction based on a description of the subject matter alone. None of the systems that we considered here have this capability, but it is partially implemented in the design of the ABC system. In particular, using ABLE, developers can provide a high-level specification of the structure of an application system's data and procedures, and CARLA will convert these into context-sensitive descriptions of these structures to be provided to students. In addition, using BILL, developers can provide a high-level description of the behavior of the application, and take advantage of that description to create any number of exercises with the application. Thus, the only instructional specifications needed for ABC are verbal descriptions of the application's components, descriptions of particular exercises needed to develop competency, and the curriculum needed to control instruction.

Programming Skill Requirements. Most of the alternatives discussed above require some programming proficiency on the part of the developer. In view of the general shortage of good programmers, especially those familiar with NOHIMS, consideration should be given to the particular skills needed to develop courseware.

Authoring Systems. Frame-based CAI is typically developed using an authoring system. Using such a system, the author details each interaction with the student in terms of a frame. Specifying a typical frame involves describing what is displayed to the student, what the student does in response to the display, and how to select the next frame on the basis of the student's responses. Since the instructional options available in frame-based CAI systems are so limited, their authoring systems are usually accessible to nonprogrammers.

BILL---The Exercise Specification Language of the ABC System. If the ABC system is adopted, those responsible for formulating exercises and examples need to be familiar with BILL. BILL is not a programming language but rather a specification language for describing the application and exercises for practice in the application.

Specifying the Application: BILL requires that the actual behavior of the application (NOHIMS) be described in sufficient detail to simulate it for the purposes of instruction. This specification need not be as detailed as the computer code that actually implements the application. Rather, BILL will provide a collection of "emulators" that allows for such a description in high-level data management terms and not at the level of actual computer code.

Describing Exercises and Examples: Once the application has been described, any number of exercises or examples can be written. Exercise specifications denote the overall goals of the exercise and instructional materials used in the course of the exercise. Details regarding behavior of the application need not be repeated in exercise descriptions. Indeed, exercise descriptions can be written by individuals other than those who write the application description.

Programming Languages. We have discussed the possibilities of providing a CAI system that is written directly in some high-level programming language, namely, MUMPS, a specialized authoring language, or some other general-purpose language. If some or all of the system is implemented using this approach, then a programming staff must be available to develop and maintain those parts. The need for such a staff lessens with the extent to which instruction can be developed using a high-level specification. Naturally, the availability of programmers for various languages and the cost of familiarizing them with NOHIMS should be considered in choosing a language.

Comparison of Alternatives

Table 1 compares the approaches discussed above in terms of the features important for any evaluation. The results of this analysis are easy to summarize. With respect to instructional effectiveness, the frame-based approaches can be eliminated because they fail to provide practice. If, in addition, the CAI system is to be fully integrated with NOHIMS' hardware and software, only the first three approaches listed in the table are viable alternatives. Further, if a major overhaul of the CAI software becomes necessary because of changes to NOHIMS, only ABC or AbC would be viable alternatives because they are generic to MUMPS application systems.

Assumptions Used in Cost Rankings

The cost side of the picture deserves a somewhat closer look. In Table 1, we have used certain principles to rank the alternatives in terms of the cost of hardware, of software development, and of maintenance. To deal with hardware costs, we assumed that the external systems would cost more than those running under NOHIMS software. Turning to software, we first assumed that hard-coded routines would be cheaper to develop but more costly to maintain than a modular, well-structured system. We also assumed that frame-based CAI is easier to develop and to maintain than other forms that offer a wider range of instruction. Finally, we assumed that it would be easier to write and maintain a CAI program for NOHIMS written from scratch in MUMPS than in any other programming language, if only because NOHIMS is written in MUMPS.

General Considerations

Because maintenance is bound to comprise the lion's share of the costs of a system of this sort, it is clear that ABC is the best alternative for offering all of the essential instructional features. And, there are other pertinent considerations that point to ABC as the best solution.

Modularity and structure are a very strong point for ABC. Good programmers are difficult to find, both inside and outside of the MUMPS community. Systems that can be developed and maintained by nonprogrammers have a clear advantage. In connection with this point, we should indicate that the low ranking given to ABC in software development (see Table 1) is based on a first-copy cost. Developing instruction for a second system based on a different application will be considerably less costly since all that needs to be written are the specifications needed by ABLE and BILL.

Table 1 - NOHIMS CAI COMPARISON CHART

CAI Approach	Integration: Compatibility of CAI Module with NOHIMS Hard/Software	Instructional Functionality	Modularity/ Operational Features	Prerequisite Skills[2]	Costs[1] H	S	M
Versions of ABC							
ABC	Hardware/ Software	Descriptive/ Practice	Student Interface Verbal Descriptions Curriculum Exercises Application Structure Application Behavior Generic to MUMPS Application Systems	BILL	1	6[3]	1
AbC	Hardware/ Software	Descriptive/ Practice	Student Interface Verbal Descriptions Curriculum Application Structure Generic to MUMPS Application Systems	None/ MUMPS for bILL	1	3	1
Conventional CAI with NOHIMS							
MUMPS	Hardware/ Software	Descriptive/ Practice	Student Interface	MUMPS	1	4	4
STUDY or DIALOG (frame-based CAI)	Hardware	Descriptive	Student Interface Verbal Descriptions Curriculum	Authoring System	1	2	3
External Conventional CAI							
Programming Language	Neither	Descriptive/ Practice	Student Interface Verbal Descriptions Curriculum	Programming	2	5	5
Frame-Based CAI	Neither	Descriptive	Student Interface Verbal Descriptions Curriculum	Authoring System	2	1	2

1 Ranked: H = Hardware, S = Software Development, M = Maintain Congruency between CAI Module and NOHIMS.

2 Skills needed beyond requisite instructional skills.

3 Includes development of the ABC system.

Another general issue that deserves some mention is the risk associated with each approach. Since ABC is a technological innovation, it naturally carries the most risk, but it is important to consider just where that risk lies. There are two relevant concerns. Will ABC be able to deal with instruction for the full range of information management systems that we envision, and will the people responsible for formulating instruction in those systems find the specification language and tools (viz., ABLE and BILL) usable? These two questions, while relevant to the general applicability of ABC, are not of much concern with respect to the NOHIMS application. Because NOHIMS is ABC's development case, NOHIMS will clearly fall within the range of ABC's applicability. By the same token, the ABC development team will make sure that they at least can use ABLE and BILL to produce a specification for NOHIMS. The major risks associated with ABC are relevant in applications beyond NOHIMS.

There are also risks associated with the alternative approaches listed in Table 1. If the CAI system is programmed in MUMPS or some other language, changes in NOHIMS and/or loss of programmers familiar with the CAI program could render the CAI system obsolete and unmaintainable. The major risk associated with the frame-based approaches is that they will be ineffective because of failure to provide guided practice in operating NOHIMS. [There is some evidence (Anderson, Boyle, & Reiser, 1985) that instruction such as that offered by ABC is about as effective as one-on-one tutoring.]

Step 2: Identify NOHIMS Functions by Class of User

There are four distinct classes of NOHIMS users, and each class uses a different constellation of the data entry and retrieval functions of NOHIMS.

Data Entry Clerk

Data entry clerks are responsible for accurately entering medical data and survey data into NOHIMS. These data are contained on standard forms. Data entry clerks must read these forms, identify appropriate codes for the data thereon, and use one of the NOHIMS data entry options to enter the codes into NOHIMS. There is a reasonable correspondence between the forms and the structure of the data entry options, but the clerk must exercise a certain amount of control over the process, for example, to effect transitions between suboptions corresponding to different parts of the form.

Occupational Health Medical Personnel

Occupational health medical personnel use NOHIMS to obtain reports of medical data. Most physicians will use NOHIMS to retrieve data on individual patients under their care. These data can be retrieved through a particular system option that can display and/or print patient data in a number of different formats. Occupational health physicians or technicians can use this capability to obtain reports that suit particular circumstances (e.g., an individual encounter report, a summary of the patient's medical record, or a flowchart tracking selected medical items).

Some physicians may also use NOHIMS to conduct miniature epidemiological investigations at their facility, perhaps triggered by what is perceived as an unusually high incidence of some complaint or disorder. NOHIMS supports this kind of investigation through a report generator in the medical component of NOHIMS and an interactive query function in the industrial component. Using the report generator and/or query function requires specifying the data items to be retrieved, the patient/worker selection criteria, and the analyses needed to summarize the data.

Industrial Hygiene/Safety Specialist

Industrial hygiene/safety specialists use NOHIMS to track and monitor occupational hazards at Navy industrial sites. Their use of NOHIMS involves configuring the industrial component of the system to their installation and maintaining that configuration. It also involves maintaining survey data on occupational health hazards and individuals' exposure to these hazards. Industrial hygiene/safety specialists also use NOHIMS to generate a number of reports, some required by law, others generated on an ad hoc basis to deal with specific situations.

Industrial hygiene/safety specialists use particular NOHIMS options to accomplish each of these functions. An editor is provided to set up and maintain configuration data about the installation. A special survey module is used to maintain survey data. Report generators are available for standard reports, and a query facility can be used to generate ad hoc reports.

NOHIMS System Manager

The NOHIMS system manager is responsible for the day-to-day operation and management of NOHIMS. This individual needs to understand how the system was initialized, how to change or update system parameters, and how to insure the integrity of the NOHIMS database. He or she maintains the security features of the system, sets up user-defined displays and flowcharts, monitors data entry procedures for accuracy, produces management reports on a periodic basis, and assists users in properly formulating ad hoc queries of the NOHIMS database. On a daily basis the system manager reviews the NOHIMS error logs to identify any program errors, disk errors, or database errors so that corrective action can be taken. This individual also regularly checks the status of "Monitor," the background "caretaker" program that secures the system against data loss, and restarts operations after a system crash. The NOHIMS system manager initiates back-up procedures on a daily basis and is responsible for periodic purging and/or archiving of data files.

Step 3: Identify Knowledge Requirements

General Considerations

In order to accomplish the functions described in Step 2, NOHIMS users need to know about the data structures present in NOHIMS, the files used to maintain data, and the procedures for data entry and retrieval. Each of these domains can be represented as a hierarchy. As Figure 1a shows, data

Figure 1a. DATA STRUCTURE (a hierarchy of data elements)

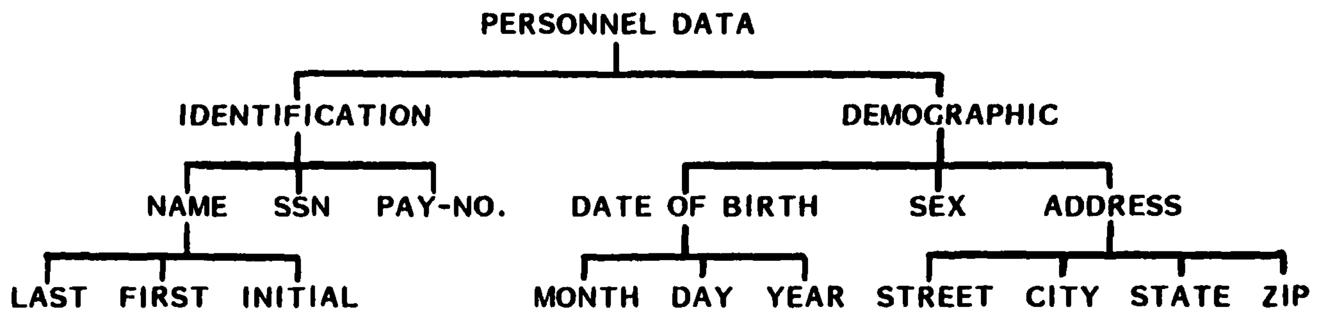


Figure 1b. FILE STRUCTURE (representative and corresponding to Figure 1a data)

FILE RETRIEVAL SUBSCRIPTS	DATA CONTENTS OF THE FILE AT THE LOCATION DENOTED BY THE SUBSCRIPTS
(Sub.1)(Sub.2)	
"IDEN",SSN	$\overbrace{\text{LAST,FIRST INITIAL}}^{\text{NAME}}$ ^ SSN ^ PAY-NO.
"DEMO",SSN	$\overbrace{\text{MONTH/DAY/YEAR}}^{\text{DATE-OF-BIRTH}}$ ^ SEX ^ $\overbrace{\text{STREET ^CITY ^STATE ^ZIP}}^{\text{ADDRESS}}$
"IDEN",PAY-NO.	SSN (Data provide pointer to above SSN data)
"IDEN",NAME	SSN ^ SSN ^ ••• ^ SSN (Data are a list of SSN pointers of all personnel having the same name)

NOTE: This file structure provides a cross-index of each person by NAME, SSN, and PAY-NO. for direct retrieval of all data, given any one of the three identification attributes.

Figure 1. Example of a Hierarchical Data Structure and Its Corresponding File Structure.

structures can be represented as a hierarchy of data elements. File structures constitute a hierarchy denoting the subscripts that index the data and the content of the data (see Figure 1b for an example). Procedures can be described as hierarchies that represent successive specializations of more general procedures to specific tasks. Figure 2 shows an example from NOHIMS. There are, of course, other knowledge requirements besides knowledge of these three hierarchies. For example, users must know the attributes of different data items and the selection conditions for procedures.

ABC is designed to teach users about hierarchies representing data structures, data files, and procedures for data entry and retrieval. It teaches knowledge of the hierarchies themselves and the particular features of the hierarchies needed for successful use of the system. (These particular features are detailed in the next step.) ABC can be configured to teach any data entry and retrieval procedure that can be represented in a general hierarchical fashion.

Knowledge Requirements of Particular Users

The knowledge requirements of individual users will vary widely among and within classes of NOHIMS users. Data entry clerks need a high level of competence in NOHIMS data entry procedures and some small knowledge about the attributes and types of data. They do not need extensive knowledge of any of the report generation and query procedures. Most physicians need to know the structure of data associated with patient records and the procedures for obtaining reports on individual patients. In addition, some physicians will need to know about the file structures in both the medical and industrial components of NOHIMS and the procedures for generating COSTAR reports. Industrial hygiene/safety specialists will need detailed knowledge of every aspect of the industrial component, but they probably will never have occasion to access the medical component directly because of the requirement for confidentiality of the patient record. In summary, ABC will operate in a situation where users need to learn about the structure of an application system's data, files, and procedures, where this knowledge can be represented as hierarchies together with certain ancillary information, and where knowledge requirements vary widely among users.

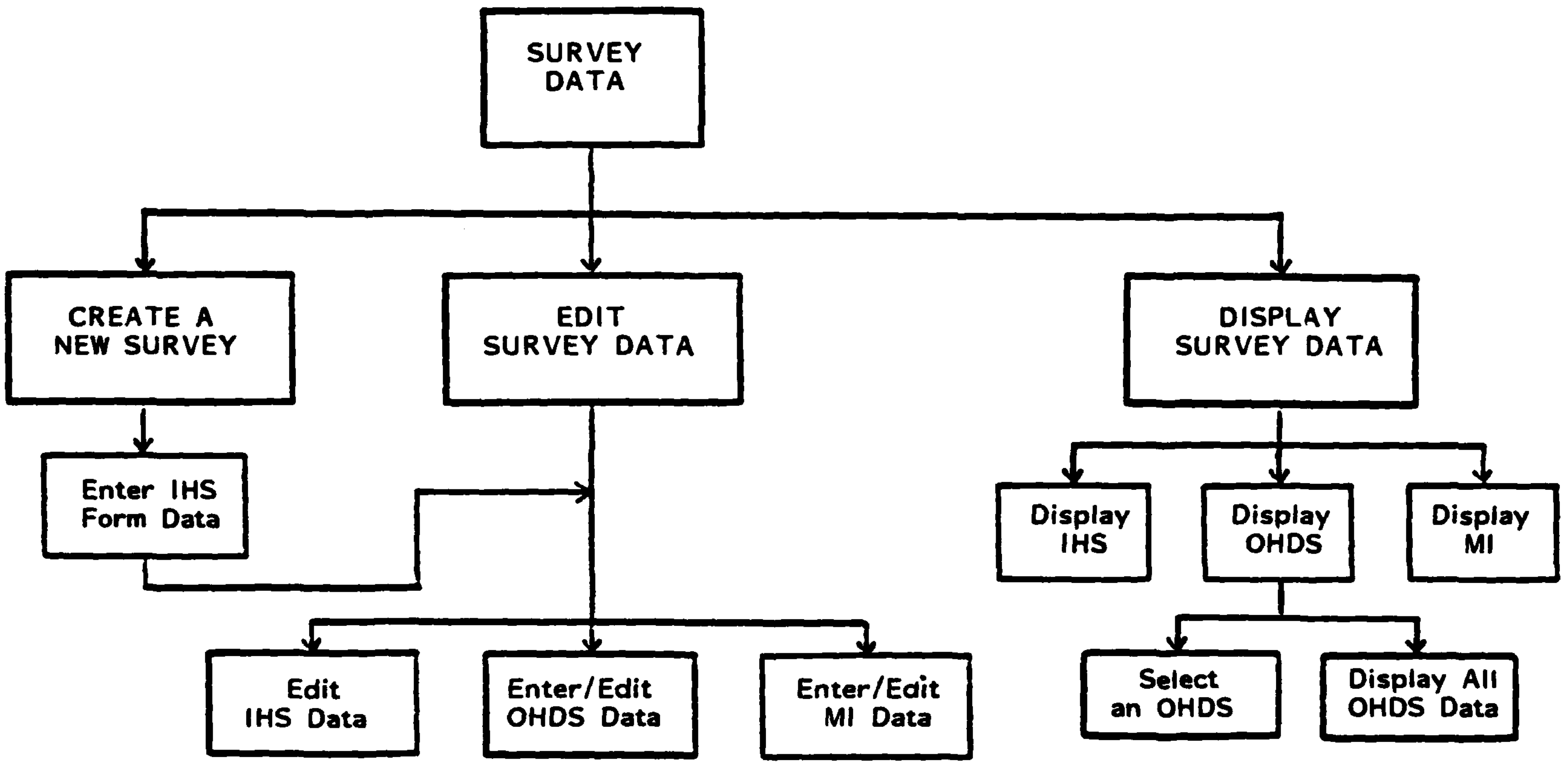
Step 4: Instructional Approach and Objectives of CAI for NOHIMS

The Training Environment

Knowledge requirements, the most important aspect of the training situation, were discussed above, but there are other relevant features of the situation that further contribute to constrain our approach.

Personnel Turnover

NOHIMS user training needs to be geared to situations with reasonably high personnel turbulence. This means that the training should be available as a turnkey system. New users should be able to easily find the training needed for their particular jobs. The training of new users should make none or few demands on the rest of the NOHIMS installation.



LEGEND:

IHS	=	Industrial Hygienist Survey Form
OHDS	=	Occupational Hazard Data Sheet
MI	=	Material Inventory

Figure 2. Example of a Procedure Hierarchy in NOHIMS.

Training Management and Certification

Some users, such as data entry clerks, will need to meet certain levels of competence before being given responsibility for "live" NOHIMS data entry. The training system should support this need through computer-managed instruction. Students should be held to a curriculum that ensures their mastery of requisite skills, and records should be kept of their performance in order to certify their competence.

The Casual User

Many NOHIMS users, and physicians in particular, will need only superficial knowledge of particular aspects of NOHIMS. We can also expect their uses of NOHIMS to vary from time to time. Therefore, a fixed curriculum provided in a single course of study would not be responsive to casual users' needs. Instead, they need training that is focused on the particular functions that they need to use and is delivered at the time of that use. High skill levels and thorough knowledge of the system are not particularly critical or appropriate for casual users.

Reference Materials

Provided with NOHIMS are three voluminous user reference manuals [NOHIMS Users' Reference Manual (Medical Component), Version 1.0, 1984; NOHIMS: Navy Occupational Health Information Management System Users Guide (Industrial Component), Version 1.0, 1985; NOHIMS System Managers' Manual, Version 1.0, 1985] that contain operating instructions for the entire system. In theory, these manuals could be used as training materials or as reference materials for casual users. But in fact, manuals of this sort are not commonly used. Their very size is intimidating. More often than not, manuals do not contain direct answers to particular questions, and, if they do, those answers are difficult to find. And, manuals are all too often removed from the physical vicinity of the terminals where they are needed. Consequently, we recommend that NOHIMS training (and operation) should be available in a paperless environment independent of external reference or training materials.

System Evolution

NOHIMS, and most other systems of the same sort, are in a continual state of evolution as features and modules are added or changed. The major impact of this fact is on the system architecture described in Step 5, but it also has training implications. In particular, users need a way of obtaining information on new features and changes as they are introduced into the system without having to retrain on the remaining aspects of the system. Ideally, the CAI system should maintain records to determine who needs training in new or different functions.

Instructional Approach

The preceding considerations and the analysis of functions and knowledge requirements discussed above indicate that NOHIMS training must have the following features in order to be successful.

Instruction on Demand

Training should be offered in the operational NOHIMS environment. That is, a NOHIMS user should be able to use training facilities without terminating his or her interaction with NOHIMS. Upon entering the CAI program, the user should be given the means for directing training to particular topics. Once training on those topics has been completed, the user should be returned to NOHIMS in the same state as he or she left it. This goal would be possible in the medical component down to a least the second level option drivers. In Version 1.0 of the industrial component, this goal is possible at all option driver levels.

Practice

One of the most important and obvious aspects of the knowledge requirements discussed above is their procedural nature. Because of this, practice should be the major component of an adequate training system. The system should offer a number of exercises on each NOHIMS procedure. Tutorial intervention should be available during the exercises. Help should be given when students ask for it or when their behavior indicates that they need it. There should be both specific ("What do I do next?") help and global ("What approach should I use?") help. Students should be able to convert exercises to worked examples by asking the CAI system to complete the exercise on its own.

Computer-Managed Instruction

Instruction should be given under the control of a curriculum. This curriculum should indicate the prerequisites for each topic or procedure and select training sequences that provide prerequisites in the correct order. The CAI system should administer tests and keep track of performance on exercises, and these data should be used to determine mastery of particular topics and procedures. The curriculum should allow for independent study of different topics by different classes of users on different occasions. The system should keep records so that managers can determine the skills that individual students' possess.

On-Line Reference Material

Students should have comprehensive reference materials available on-line during both their interaction with NOHIMS and the CAI system. The reference materials should describe the structure and content of NOHIMS' data, files, and procedures. Users should be able to enter free-form textual inquiries to the reference system. The reference system, in turn, using inferential techniques to resolve conflicts or ambiguities in the user's input, should be able to construct an option menu of candidate subjects or relevant suggestions for the user's selection. Students should be able to browse the hierarchies describing the structure of NOHIMS' data, files, and procedures. The reference system should be integrated with the curriculum so that students can initiate practice on any topic from the reference section, and students should be able to make use of the reference facility during exercises.

Step 5: CAI System Architecture

The Need for a Generic System

As we mentioned previously, a major design goal of the CAI system is its applicability to a wide range of system applications. We adopted this goal for several reasons. First, we believe that CAI authors should not have to rewrite CAI software to accommodate changes in the material being taught. Second, it should not be necessary to write new CAI software for distinct but similar applications. Third, a CAI system should, as much as possible, automate the formulation of instructional procedures, leaving it to subject matter experts to specify the content of the instruction in its own terms (versus computer code).

This said, we need to be specific about the generality of the system and the form of the content specification. ABC offers training in the use of data entry and retrieval systems. ABC can be configured as a CAI system for any such application system (written in MUMPS) that meets certain structural requirements and certain functional requirements for interacting with the user and the database.

Structural Requirements

It should be possible to characterize the application in terms of three structural hierarchies: one describing the data structures used in the system, another describing the file structures used in the system, and a third describing the procedures used to operate the system.

The data structures are simply tree structures in which nodes are primitive data types or data structures themselves. Any node can be defined in terms of some permissible transformation on some other data structure.

A file structure is a series of location-content pairs. The location part of each pair is a series of primitive data types representing subscripts. The content part can be any data structure. Files can be defined in terms of general file types, and it is possible to cross-reference common fields in a different or the same file.

The procedure hierarchy is also a tree structure. Its root refers to the application itself and descendants represent specializations of more general procedures. The procedure tree will often map into the tree structure of a menu system used to control the application. Any node in the procedure tree can represent a common procedure that may be used in different places in the tree.

Functional Requirements

To allow for proper conduct of exercises, the CAI system must be able to simulate relevant aspects of the application system's behavior in a way that allows for tutorial interaction. We plan to implement this simulation using a standard set of primitives, called emulators, that describe how the application interacts with the user and with its databases. Thus, if instruction calls for exercises with a certain function in the application, the behavior of that function must be describable by the emulators provided with ABC.

ABC

As explained earlier, ABC consists of three executable components. One component, ABLE, is used to specify the structures (data, file, and procedure) for the application, descriptive information needed for instruction, and curriculum structures. The second component, BILL, is used to specify how the system behaves and to define the content of particular exercises. The third component, CARLA, delivers instruction in the application's environment according to the specifications provided through ABLE and BILL. CARLA is the only component that needs to run in the application system's environment. Students using CARLA can identify what they want to learn and use the training regimes specified in ABLE and BILL to receive appropriate training or to retrieve appropriate reference material. A more detailed description of these three components is presented in the next section.

Description of the ABC System

The ABC system consists of three major executable components named ABLE, BILL, and CARLA. The design requirements for each of these three components are described below.

ABLE

ABLE is used by the authors of the CAI module to provide a structured description of the application for reference use. The definition includes all data items, file structures, and operational procedures that exist within the application being taught (in this case, NOHIMS). The ABLE component can be segmented into three major areas: the Data Definition Segment, the File Definition Segment, and the Procedure Definition Segment.

Data Definition Segment

The data definition segment produces a catalog of information on data groups and data items as they exist in the application system. This catalog is not a representation of the content of data files for the application. Rather, it provides a description of the data groups, item names, and item construction attributes independent of where or how the data are actually used. The other two definition segments of ABLE depict usage, arrangement, and file location of the data groups and items.

Data Definition Structure

Data are defined within multilevel structures. Entries at the first level will usually be a data group such as "PERSONNEL DEMOGRAPHIC." Each such first-level entry constitutes a "new" data set. Under each first-level entry a variety of associated data can be defined as either a subordinate, descendant, data group, or as a data item. A data item is an entry which cannot be further reduced and does not have any subordinate entries. A data group is any entry having one or more descendant entries. Any descendant entry can consist of either subordinate groups and/or data item entries. Any first-level entry can be a data item entry. The depth of descendant definition is not limited.

Each application-specific data group or item that is to be usable for any ABC purpose must be given an "external" name and an "internal" name. A data group or item that is created and used only as a component of the ABC operation need not have an external name. External names will usually consist of the word or phrase that is used to identify the data in the context of the application system. ABLE also allows any number of synonymous names to be defined for any entry. The synonymous name would normally be variations of the primary data name as might commonly be used by personnel who utilize the system. The internal name is mandatory only if the data entry is to be used in a training curriculum exercise, contains instructional text or example material which may be referenced during a curriculum exercise, or is an ABC dedicated entry. The internal name used can be either the code that is created as an identifier for the ABC directory, or it can be a unique author-specified configuration. The internal name uniquely identifies a data group or item and is used within the ABC components as both a reference and as the variable that contains or is to contain the current value of the data. These names, both external and internal, are filed in appropriate regions of the ABC directory with coded pointers to the data definition information.

For each named data group or individual data item, an instructional text display and example set may be entered and associated with the named data. The instructional material is entered via the ABC text editor and may include example display and any amount of text.

Redefinition of Data Entries

There are many circumstances when either data must be transformed or components of a data item are to be used for other purposes. For example, the name of a person may be defined to be input as a single data item having a specific format. However, because directory filing conventions for a person's name consists of using the last name and the first name as separate identifying subscripts, the name must be redefined so that a data item exists for not only the whole name, but also for each component of the name. ABLE allows this type of redefinition at any level. Also, any entry can be redefined any number of times to accommodate a variety of required transformations of the same data entry.

Descriptive Attributes and Data Transformations

All data item entries which will be invoked during any ABC exercise must have some operational attributes defined. Information concerning data type, required transformations, format, and entry editing of the data item as may be applicable for use of the data in ABC practice scenarios is specified at the data item level of entries. The extent of attribute definition depends upon the intended ABC use of the data group or item. That is, if an item is to be input by the student for the sole purpose of training the student in the correct entry configuration of the item, then only entry editing attributes need to be defined for the item. On the other hand, if the item will be used as a filed reference to other data or exercise steps, then any translation or data transformation process required to facilitate filing and retrieval of the data must also be defined. Data group entries do not have attribute information. Attribute information appears only on the lowest level entries, which are the data item entries.

Data transformations will usually be accomplished automatically within the ABC components whenever the data are input, used, or output. ABLE allows construction of required conversion tables for any type of code-to-value transformation. For example, the input choices and output display for data item "SEX" are "MALE" or "FEMALE." However, the "SEX" data item is to be stored as "M" or "F." ABLE easily facilitates such transformations. Some forms of data transformation are far more complex than such simple conversions. Therefore, the ABC task execution language may be used in these situations to construct an executable data transformation task when none of the various intrinsic data conversion methods within ABC are applicable.

Data item input editing is normally an inclusive portion of the input emulator prompting routine that is designated as the input method to be utilized for soliciting the item. However, an ABC editing task can also be created if necessary.

A primary attribute for each data item entry is the data type. This attribute describes the general characteristics of the item and normally denotes the methodology that ABC will employ during input, output, storage, and internal manipulation of the data. The intrinsic ABC data types are as follows:

TYPE:	USAGE
	(NOTE: All "Alpha" type values are less than 81 characters.)
AC:	An alpha constant value ("a").
AL:	An alpha value (a).
AR:	Multiple alpha values having a specific separation character sequence (i.e., a1-a2).
AS:	An alpha value string (a,a,...,a) where each a translates to a specific list value.
AV:	An alpha value which is to transformed via an ABC task.
AX:	A single alpha value to be translated to a specific list value.

(Continued)

CA:	An ABC task-constructed alpha value (a).
CD:	An ABC task-constructed date (any resulting format).
CN:	An ABC task-constructed integer value (n).
CP:	A multiple value ABC construction where values are separated by a specific character (v^v^...^v).
CR:	An ABC task-constructed decimal value (r).
CT:	An ABC task-constructed multiple line free text.

DD:	A dd/mm/yy date value.
DE:	A dd mmm yy date value.
DH:	A MUMPS $H date value.
DJ:	A yyddd Julian date value.
DM:	An mm/dd/yy date value.
DV:	A date value to be transformed via an ABC task.
DX:	An inverted MUMPS $H value (99999-$H).

NC:	An integer constant value (#).
NI:	Single integer value (n).
NR:	Multiple integer values having a specific separation character sequence (i.e., n1/n2).
NS:	Integer value string (n,n,...,n) where each n translates to a specific list value (n <-> value).
NT:	Single integer value followed by a limited length text (n text).
NU:	Single integer value followed by a unit value (n unit).
NV:	An integer value which is to be transformed via an ABC task (n <-ABC-> value).
NX:	A single integer value (n) to be translated to a specific list value (n <-> value).
NY:	An integer value followed by a limited text (n text), the integer portion of which is to be translated to a specific list value (n text <-> value text).

RC:	A decimal constant value (#).
RI:	Single decimal value (r).
RR:	Multiple decimal values having a specific separation character sequence (i.e., r1/r2).
RT:	Single decimal value followed by a limited length text (r text).
RU:	Single decimal value followed by a unit value (r unit).
RV:	A decimal value (r) which is to be transformed via an ABC task (r <-ABC-> value).
RX:	A single decimal value (r) to be translated to a specific list value (r <-> value).

TL:	A multiple line free text.

Other attributes such as minimum and maximum text length, value ranges, value selection and transformation tables, and other special handling tasks can be specified in the entry.

File Definition Segment

The ABC files are not designed to represent the application system's file construction or content. ABC files are designed to contain and facilitate storage and retrieval of data that are either predefined, entered, or constructed during the ABC exercise training sessions. Also, ABC contains executable code for a variety of commonly used sequential and random access storage, retrieval, and selection methodologies. Any method code set can be invoked for any ABC file having the appropriate design. By proper file definition and selection of the ABC intrinsic file handling methods, practically any data retrieval action native to a production application system can be emulated.

File Structure

Each ABC file is given a descriptive name and a 2-to-4 character internal file name, both selected by the author. The file name is selected so as not to conflict with any existing application files. For each file, the definition will consist of a series of file node descriptors. These specifications are arranged in pair sets of data item internal names. One set defines those items that are used as node subscripts, which denotes the node location within the file. The other set defines the data items which denote arrangement and content of data stored at the node. There can be one to several subscript items specified, each of which can be an integer or decimal value, a limited length alpha value, or an ABC constructed value. The ABLE definition process will not allow any invalid data definition item to be used in the specification of node subscripts, nor will ABLE allow an invalid aggregate subscript configuration to be defined. All ABC files are of the hierarchical design. That is, each additional subscript for a node constructs nodes which are descendant nodes of the previous subscript. Also, a file node can be defined without a data content definition.

Intrinsic File Definitions

As mentioned above, ABLE contains many intrinsic storage and retrieval methods. For each such method, there exists "skeleton" file structures. To define a file using these skeletons, a specific file "method" is specified. ABLE will thereafter solicit the data item names to be used for applicable method subscripts and allow node content definition where the node data content is not determined by the methodology in use.

Cross-Referenced File Nodes

During the definition of files, whenever the same data item name is used as a subscript in either different files or different node subscript positions of the same file, a bi-directional cross-reference is intrinsically defined and noted by ABLE in a cross-reference table. This feature maps and facilitates emulation of any existing application data relationship or retrieval aspect.

Procedure Definition Segment

This segment of ABLE allows the author to define a hierarchical map for the operational action of the application system. The hierarchically structured procedure definition is initiated by definition of the major functional processes within the application system. Each major function is then elaborated in a hierarchically descending expansion of the structure until every procedure that the student should be able to reference is identified.

Procedure Identification

At each procedure node of the structure, the author provides an external name and a unique internal name. The external name will normally be the same terminology as used in the application system. As with the external data definition names, synonymous name descriptions are also allowed to be defined for any procedure. These names are also in the ABC directory in the same fashion as data names. However, they have pointers to the specific procedure. Duplication of a data name and a procedure name is not an uncommon event. This will not be a problem, however, because ABC will recognize that both data and procedures exist with the same name and either select the appropriate one in the context of the current task or ask for a selection of "data" or "procedure" to be performed. In the case of student query response, both references to data and procedure may be allowed. The internal procedure names are only used within the internal ABC definition and operations. For each procedure, a general educational description text can be created. Again, like the data definitions, this text will be available for display when identified by student selection of the procedure name.

In many cases during the actual operation of the application system, a specific procedure or procedure set will be invoked from various other prodecures. An example would be a person name look-up procedure that is used for selection of a target person for several related or disassociated tasks. To avoid repetitive definition of identical procedures, the author may reference any procedure, via the use of an existing or proposed internal procedure name, from within the definition of any other. Also, different external names and/or instructional material may be defined for identical procedures where the use of the procedure in the application system may be different. This capability can be used to accurately emulate "on-the-fly" operations that exist within some applications.

Exercise Curricula

Another optional attribute of the procedure definition allows the author to indicate an intent to associate the procedure with an executable training lesson. These lessons are constructed using the BILL component of ABC and are invoked and executed by the student interface component, CARLA. The author need not indicate the existence of a lesson for the procedure during the procedure definition, but can indicate an intention to create such a lesson. Whenever BILL is used to create lessons, the procedure map produced by ABLE is used to identify the subject lesson-procedure association or remind the author of the need to construct the lesson. BILL will automatically set the ABLE lesson attribute to reflect the status of any lesson that is produced.

In general, any data, file, or procedure definition of ABLE can be edited or deleted, or new definitions added by the author at any time. If such new ABLE definition or alteration affects existing operational aspects of other ABC modules so as to render them inoperable, the related areas will be identified and logged as operational conflicts. The status of the affected areas will remain "inoperable" until the author resolves the conflicts. This type of conflict monitor is typical of definition alterations performed in all ABC components and aids the author in the coordination of the various aspects of the CAI design.

BILL

Bill is an authoring tool used to specify the exercises that students use to practice with the application. Exercises, by their nature, involve use of that application, so CARLA, in order to deliver exercises must not only know the content of particular exercises but also how the application functions in general. BILL is configured so that an author/subject matter expert can provide a detailed specification of the application's behavior and use that specification to create any number of exercises for any number of lessons.

Simulating the Application

During exercises, students will work with a simulation of the application and special instructional versions of its databases. The author must therefore detail the behavior of the simulation and the content of the instructional databases. Since the latter can be specified in a straightforward way using the structures defined in ABLE, it is the simulator that concerns us here.

The simulation of a particular application will be written in terms of a transition network. This network will have four parts:

1. a set of states that correspond to procedures in ABLE's procedure network,
2. the conditions that determine transitions among the states,
3. side effects that occur as the consequence of transitions between states, and
4. local variables that contain control information not captured as state information.

Both the conditions and side effects are specified using the emulators mentioned above. An emulator is a precoded operation that works in conjunction with certain arguments or variables. These arguments can be the local control variables mentioned above, functions on the instructional database, or input from the student-user. Emulators will be available for most commonly used data input prompting methods, random access data storage, subject identification, selection, and retrieval actions. There will also be emulators for utility operations such as date format conversion and many other data translators, text input, editing, parsing, storage, subject or content retrieval tasks, candidate list preparation for a variety of specific or ambiguous data identifiers, output display construction, and menu option drivers. Where no emulator is applicable for a desired effect, a task language can be used to produce one.

Lessons and Exercises

BILL is a lesson authoring tool. The primary function of a lesson is to allow users to practice with the target application in a tutorial environment, but lessons may also include ancillary material to explain the objectives of exercises, test the student's mastery of the lesson objectives, and provide additional instruction relevant to the exercises.

On one level, BILL is a frame-based CAI authoring system with two basic types of frames---display frames and exercise frames. Display frames display textual information and accept student input. Authors can specify how to test student input and how to branch depending on the result of such tests. Display frames will be used to introduce exercises, allow students to choose among them, and to test students' knowledge between or after exercises.

The other kind of frame is more sophisticated. When an exercise frame is invoked, the student is placed in a working simulation of the target system, and must accomplish a particular task using that simulation. An exercise frame defines an interaction between the simulator and the student. At a minimum it must specify the start state and a goal state (taken from the state transition network). In addition, it will often specify essential intermediate states on the solution path and constraints on some of the arguments used by the emulators. Other aspects of the frame are a verbal description of the exercise and list of common mistakes.

CARLA

CARLA is the ABC system-to-student interface component of the CAI module. It is the only ABC component that operates with personnel other than the CAI authors. It is also the only component that needs to be resident on the computer system once preparation of the CAI curriculum, teaching examples, and practice exercises are completed. All information and executable CAI routines that were defined or created by both the ABLE and BILL components are contained either in CARLA or in the ABC directory or practice files.

Modes of Operation

The first interaction of ABC with a user (student) is the customary entry of an authorization code. In ABC, this is either the same authorization code that is used by the person as entry authorization to the live application system, or it is an ABC "training" authorization which has been assigned to a student for ABC use only. The normal process by which the user initially communicates with ABC is by student selection from progressive menu option displays. This "menu" mode is one of two modes of ABC operation that may be selected after student identification and is referred to as the ABC mode selection menu. It is from the selection of the "menu" option that the identification of the target training subject begins.

Menu Mode

The options that appear on the first subject selection menu are the major application functions. These are the first level application procedures that were defined in ABLE. After selection of the major function, the subsequent menu option presentation will follow the application procedure paths for that function until the target procedure is indicated as a menu option. At this point, the student will have a choice of reviewing explanatory material and/or invoking an interactive practice exercise as may be available for the subject. The student also has the option of producing a display which maps the application subprocedures of any current menu option procedure. This not only gives the student a "look-ahead" menu capability but also provides an overview of any portion of the entire application process. The student may move forward or backward in the menu presentation at will. Whenever a training exercise, procedure map, or explanatory display is either completed or exited, the student will be returned to the same ABC menu that invoked the option. At any menu selection point, the student may elect to return to the initial ABC mode selection menu.

The menu selection process is a simple interaction that is easily negotiated by even a complete novice to the computer-to-person interface. The menu selection process does not require the student to possess any apriori knowledge of either ABC usage or the application system's procedures or terminology. It is, therefore, the most effective and foolproof method of finding the target training curriculum for any given application procedure and invoking it.

Reference Mode

Presentation of the application-oriented training curricula is only one goal of the ABC system. The second goal of ABC is to be able to quickly supply any user with all instructional material available in the ABC system in an ad hoc fashion given a subject reference. This is the "reference" mode of ABC operation and can be selected at the ABC mode menu. The design of this operative mode of ABC is such that it can also be invoked from within any point of a production system without adversely affecting the ongoing application process. ABC will return the user to the production process at the exact point of departure. This production system ABC reference capability would require alteration of applicable production routines to the extent of simple insertion of a call to the CARLA reference mode entry routine.

The reference mode of CARLA operates by soliciting an ad hoc entry from the student of the subject of interest. The entry can encompass any application process, data group, or data item subject. CARLA will then attempt to identify the target subject from this entry, after which it will retrieve any descriptive or example instructional material available and present the student with menu form options to select and display the material. If a procedure is identified as a subject of interest, an option to construct a procedure map, as described for the menu mode, will be included. Also, if there are practice exercises available for the procedure, this will be noted. If the subject of interest is a group of data or a specific data item, CARLA will include an option to display all procedures known to either input or output the data. If the data are used in the application system as a subject retrieval index or have other important application attributes, they will also be noted for the data.

Very often, the student is not familiar with the correct terminology used in the application system for either procedures or data. Most subject retrieval methods depend upon entry of exact syntax in terms of usage, spelling, and phrasing of the descriptive wording. In well-designed systems, however, procedures and data are usually identified with terminology descriptive of ordinary use or functionality. CARLA will use inferential methodology to resolve conflicts or ambiguities encountered in the subject entry and construct an option menu of either candidate subjects or suggestions that are relevant to the suspected intention of the student. Some of these methods are recognition of singular and plural forms of words, spelling correction, synonym substitution, phrase reversal, and identification of extraneous syntax constituents. When constructing subject menu options from an ambiguous entry, it is much more likely that CARLA will include more subjects than intended rather than fail in the attempt to find any intended subject material.

Practice Exercise Operation

The Course of Lessons and Exercises

Once a student chooses a lesson, control will be turned over to the frame-based CAI system described in connection with BILL. The typical lesson will begin with some instructional frames that will introduce an exercise, give the exercise to the student, and, when the exercise is completed, record that the exercise was completed successfully.

As was mentioned earlier, during the exercise students will use a simulation of the application that is augmented with certain tutorial features. One of these will be a student option to review the problem or objective of the exercise. CARLA will also provide two types of help: a suggestion concerning the next step in the exercise and a global view listing crucial intermediate steps on the path to a solution. In addition, CARLA will be able to detect when the student makes any of a number of common errors or strays very far from the solution path. On these occasions, CARLA will intervene with remedial advice. Finally, students will have the option of asking CARLA to demonstrate how the exercise should be done, thus converting the exercise into a worked example.

CARLA will invoke the practice training exercise for a selected procedure at the curriculum goal for the procedure or the first higher level goal that has prerequisite goal requirements. If more than one curriculum uses the subject procedure, CARLA will construct a curriculum option menu which will explain the purpose of each applicable curriculum and allow the student to choose the one to be invoked. CARLA will then select the curriculum goal that is relative to the subject procedure. If this goal is not the starting point of a curriculum, the student will be given the option of beginning at the starting point.

The exercise session with the student will proceed goal by goal along the curriculum-prescribed goal pathway. As each goal is successfully completed by the student, CARLA will note the goal completion in a curriculum audit file under the student's I.D. code. This feature allows inspection of student training progress.

The design of CARLA allows the student to either back up through the curriculum or to interrupt the exercise session. It is expected that the CAI system will be used in a production environment as well as in a dedicated training environment. Therefore, the normal work demands of an employee may not allow an entire curriculum to be completed in a single session. When such an interruption occurs, CARLA will internally note the curriculum and last successful goal completion point for the student. When the student next enters ABC, an option to continue the curriculum from that point will be included in the initial option menu so that the student will be reminded of the pending curriculum and be allowed to continue it.

CARLA also allows "review" of completed goals by allowing the student to back up through them and review both the process and results that occurred. CARLA does not require that the exercises be re-executed by the student. The student may, however, elect to re-invoke a curriculum at any previously completed goal rather than continue through the unexercised portion of the curriculum. This capability allows a student to repeat any specific exercise or goal until satisfied with the results.

ABC Practice Files

The ABC practice files content were initially set to certain values which are utilized during execution of the exercise goals. The files consist of either static or dynamic areas. The dynamic areas contain structures and data which can be altered while the static area data are constant. A new dynamic ABC file area is initialized for each new student. This is the only area that can be altered for that student. During some exercises, it may be desirable that student input data actually be added to the initial file contents so that a descendant curriculum goal may access these new data. CARLA will allow student data alteration of files to remain in effect until recognizing a mandatory file area "reset" request from within a goal, or the curriculum is started at the primary entry point. This data retention capability allows ABC authors to give the student freedom of data input selection in cases where it is advantageous to training.

Demonstration of ABC Design Concepts

A review of work accomplished during Phase I of Contract N00014-85-C-0813 was held on January 16, 1986 at the Naval Health Research Center in San Diego, California. As part of this review, a demonstration of key design concepts in the ABC system was presented. This demonstration was made on a Tandy 2000 microprocessor especially programmed in MUMPS for the presentation. The functionality of the ABLE and CARLA components of the ABC system was illustrated along with several practice exercises and reference features.

Attending the demonstration were the following individuals.

CDR Patrick A. Truman
Naval Medical Research and
Development Command
Bethesda, Maryland

CDR James W. Allen
Navy Environmental Health Center
Norfolk, Virginia

LCDR Hank Spolnicki
Uniformed School of Health Sciences
Bethesda, Maryland

E. K. Eric Gunderson, Ph.D.
Naval Health Research Center
San Diego, California

William M. Pugh
Naval Health Research Center
San Diego, California

Diane M. Ramsey-Klee, Ph.D.
R-K Research and System Design
Malibu, California

Donald D. Beck
R-K Research and System Design
Malibu, California

Henry M. Halff, Ph.D.
Halff Resources, Inc.
Arlington, Virginia

The methodology we have designed for implementing the NOHIMS CAI module is admittedly new. However, we feel that the demonstration presented in January 1986 proves the feasibility and viability of this approach and supports our contention that pursuing this methodology will lead to a successful outcome. In our opinion, the methodology has to be new because none of the existing CAI software, either those packages written in MUMPS or in other languages, provides the adaptable, generic capability needed to develop modifiable, interactive instructional software. Development of the ABC system to meet this need has the potential to impact importantly on how training for MUMPS-based systems will be delivered in the future.

Bibliography

Anderson, J.A., Boyle, F.C., & Reiser, B.J. Intelligent tutoring systems. Science, 1985, 228, 456-462.

Avner, A., Smith, S., & Tenczar, P. CBI authoring tools: Effects on productivity and quality. Journal of Computer-Based Instruction, 1984, 11, 85-89.

Halff, H.M. Instructional applications of artificial intelligence. Educational Leadership, 1986, 43(6), 24-31.

Halff, H.M., Hollan, J.D., & Hutchins, E. Cognitive science and military training. American Psychologist, (in press).

Merrill, M.D. Where is the authoring in authoring systems? Journal of Computer-Based Instruction, 1985, 12, 90-96.

Munnecke, T., & Pettis, J.L. Evolution of the MUMPS language - Saying more with less. Proceedings of the Sixteenth Annual Hawaii International Conference on System Sciences, 1983, 343-350.

NOHIMS: Navy Occupational Health Information Management System Users Guide (Industrial Component), Version 1.0. Naval Health Research Center, San Diego, California, September 1985.

NOHIMS System Managers' Manual, Version 1.0. Naval Health Research Center, San Diego, California, March 1985.

NOHIMS Users' Reference Manual (Medical Component), Version 1.0. Naval Health Research Center, San Diego, California, December 1984.

Sleeman, D., & Brown, J.S. (Eds.). Intelligent tutoring systems. London: Academic Press, 1982.

Individuals Who Worked on the Contract during Phase I

During Phase I of Contract N00014-85-C-0813, the following individuals contributed to the successful completion of the work.

Donald D. Beck

Henry M. Halff, Ph.D.

Harry G. Klee

Diane M. Ramsey-Klee, Ph.D.

END
DTIC
1-86